Date=14/10/2020
Lecture By=Manish Mahant
Subject ⇒Revision Classes

Overview For The Lecture

Flexbox in CSS

The Flexible Box Module, usually referred to as flexbox, was designed as a one-dimensional layout model, and as a method that could offer space distribution between items in an interface and powerful alignment capabilities.

The two axes of flexbox

When working with flexbox you need to think in terms of two axes — the main axis and the cross axis. The main axis is defined by the `flex-direction` property, and the cross axis runs perpendicular to it. Everything we do with flexbox refers back to these axes, so it is worth understanding how they work from the outset.

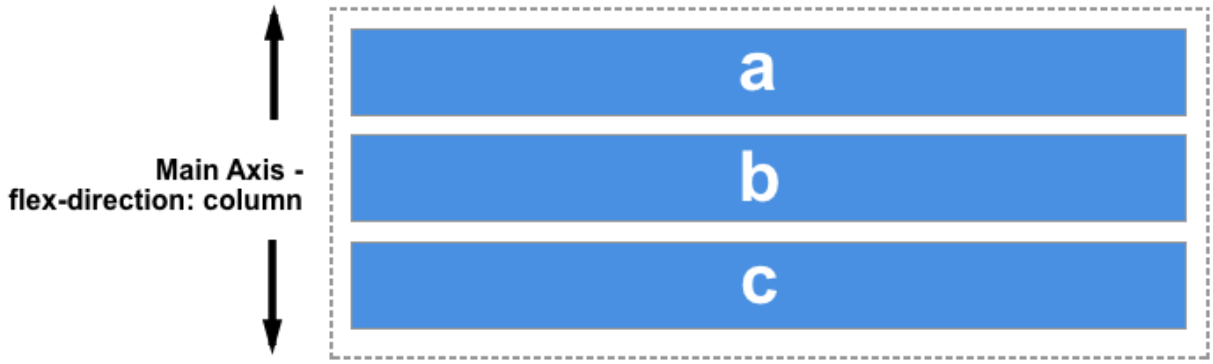
The main axis

The main axis is defined by `flex-direction`, which has four possible values:

- `row`
- `row-reverse`
- `column`
- `column-reverse`

Should you choose `row` or `row-reverse`, your main axis will run along the row in the **inline direction**.

Choose `column` or `column-reverse` and your main axis will run from the top of the page to the bottom — in the **block direction**.



The flex container

An area of a document laid out using flexbox is called a **flex container**. To create a flex container, we set the value of the area's container's `display` property to `flex` or `inline-flex`. As soon as we do this the direct children of that container become **flex items**. As with all properties in CSS, some initial values are defined, so when creating a flex container all of the contained flex items will behave in the following way.

- Items display in a row (the `flex-direction` property's default is `row`).
- The items start from the start edge of the main axis.
- The items do not stretch on the main dimension, but can shrink.
- The items will stretch to fill the size of the cross axis.
- The `flex-basis` property is set to `auto`.
- The `flex-wrap` property is set to `nowrap`.

The result of this is that your items will all line up in a row, using the size of the content as their size in the main axis. If there are more items than can fit in the container, they will not wrap but will instead overflow. If some items are taller than others, all items will stretch along the cross axis to fill its full size.

You can see in the live example below how this looks. Try editing the items or adding additional items in order to test the initial behavior of flexbox.


```
.box {
  display: flex;
}
```

```
<div class="box">
  <div>One</div>
  <div>Two</div>
  <div>Three
      <br>has
      <br>extra
      <br>text
  </div>
</div>
```

Changing flex-direction

Adding the `flex-direction` property to the flex container allows us to change the direction in which our flex items display. Setting `flex-direction: row-reverse` will keep the items displaying along the row, however the start and end lines are switched.

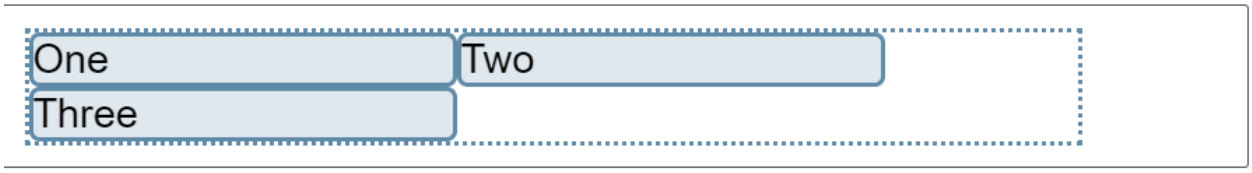
If we change `flex-direction` to `column` the main axis switches and our items now display in a column. Set `column-reverse` and the start and end lines are again switched.

The live example below has `flex-direction` set to `row-reverse`. Try the other values — `row`, `column` and `column-reverse` — to see what happens to the content.


```
.box {
  display: flex;
  flex-direction: row-reverse;
}
```

```
<div class="box">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
</div>
```

Multi-line flex containers with flex-wrap



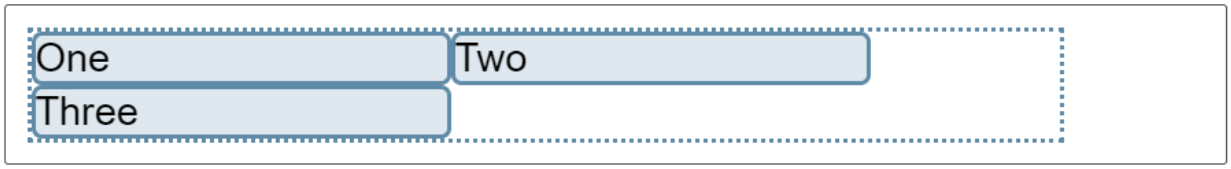

```
.box {
  display: flex;
  flex-wrap: wrap;
}
```

```
<div class="box">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
</div>
```

The flex-flow shorthand

You can combine the two properties `flex-direction` and `flex-wrap` into the `flex-flow` shorthand. The first value specified is `flex-direction` and the second value is `flex-wrap`.

In the live example below try changing the first value to one of the allowable values for `flex-direction` - `row`, `row-reverse`, `column` or `column-reverse`, and also change the second to `wrap` and `nowrap`.




```
.box {
  display: flex;
  flex-flow: row wrap;
}
```

```
<div class="box">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
</div>
```

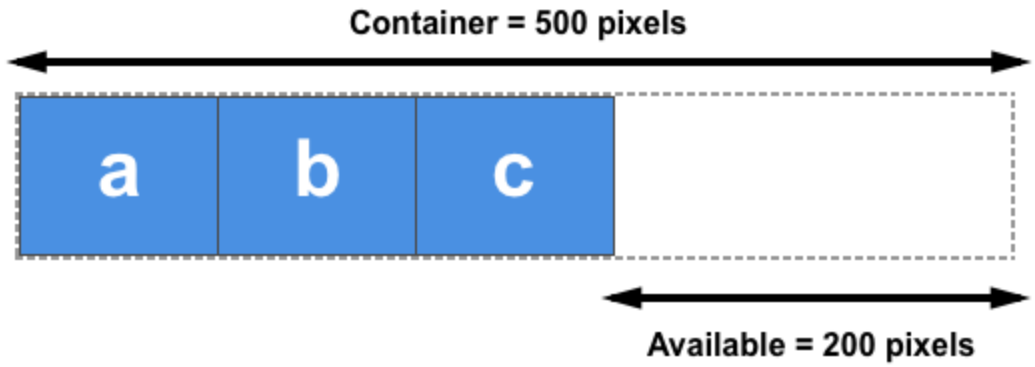
Properties applied to flex items

To have more control over flex items we can target them directly. We do this by way of three properties:

- `flex-grow`
- `flex-shrink`
- `flex-basis`

We will take a brief look at these properties in this overview, and you can gain a fuller understanding in the guide Controlling Ratios of Flex Items on the Main Axis.

Before we can make sense of these properties we need to consider the concept of **available space**. What we are doing when we change the value of these flex properties is to change the way that available space is distributed amongst our items. This concept of available space is also important when we come to look at aligning items.



There are also some predefined shorthand values which cover most of the use cases. You will often see these used in tutorials, and in many cases these are all you will need to use. The predefined values are as follows:

- `flex: initial`
- `flex: auto`
- `flex: none`
- `flex: <positive-number>`

Setting `flex: initial` resets the item to the initial values of Flexbox. This is the same as `flex: 0 1 auto`. In this case the value of `flex-grow` is 0, so items will not grow larger than their `flex-basis` size. The value of `flex-shrink` is 1, so items can shrink if they need to rather than overflowing. The value of `flex-basis` is `auto`. Items

will either use any size set on the item in the main dimension, or they will get their size from the content size.


```
.box {
  display: flex;
}

.one {
  flex: 1;
}

.two {
  flex: 1;
}

.three {
  flex: 1;
}
```

```
<div class="box">
  <div class="one">One</div>
  <div class="two">Two</div>
  <div class="three">Three</div>
</div>
```

Alignment, justification and distribution of free space between items

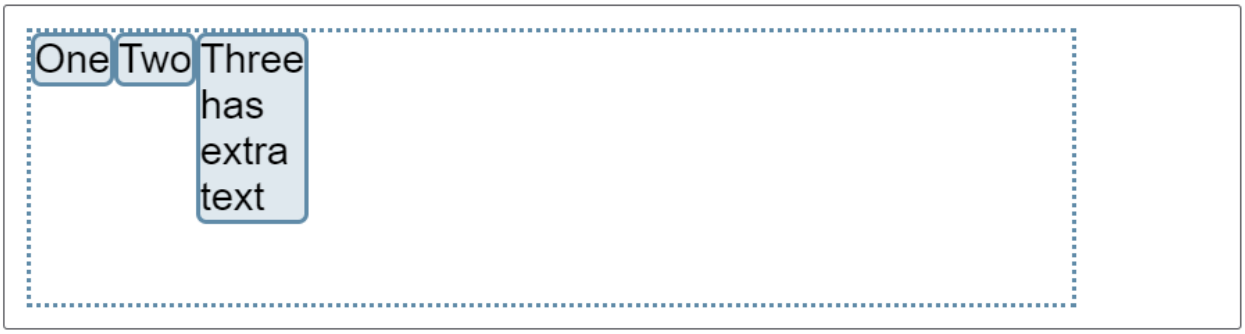
Align-items

The `align-items` property will align the items on the cross axis.

The initial value for this property is `stretch` and this is why flex items stretch to the height of the tallest one by default. They are in fact stretching to fill the flex container — the tallest item is defining the height of that.

You could instead set `align-items` to `flex-start` in order to make the items line up at the start of the flex container, `flex-end` to align them to the end, or `center` to align them in the centre. Try this in the live example — I have given the flex container a height in order that you can see how the items can be moved around inside the container. See what happens if you set the value of align-items to:

- `stretch`
- `flex-start`
- `flex-end`
- `center`




```
.box {
  display: flex;
  align-items: flex-start;
}
```

```
<div class="box">
  <div>One</div>
  <div>Two</div>
  <div>Three
      <br>has
      <br>extra
      <br>text
  </div>
</div>
```

Justify-content

The `justify-content` property is used to align the items on the main axis, the direction in which `flex-direction` has set the flow. The initial value is `flex-start` which will line the items up at the start edge of the container, but you could also set the value to `flex-end` to line them up at the end, or `center` to line them up in the centre.

You can also use the value `space-between` to take all the spare space after the items have been laid out, and share it out evenly between the items so there will be an equal amount of space between each item. To cause an equal amount of space on the right and left of each item use the value `space-around`. With `space-around`, items have a half-size space on either end. Or, to cause items to have equal space around them use the value `space-evenly`. With `space-evenly`, items have a full-size space on either end.

Try the following values of `justify-content` in the live example:

- `flex-start`
- `flex-end`
- `center`
- `space-around`
- `space-between`
- `space-evenly`
-


```
.box {
  display: flex;
  justify-content: flex-start;
}
```

```
<div class="box">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
</div>
```

Questions For Self Practice / Assignment For the Day

https://au-assignment.s3.ap-south-1.amazonaws.com/Week_16_Day_3_Assignment-2fa64023-e466-4e0f-8b5a-3071130a307c.pdf

Resources For The Lecture

https://www.w3schools.com/css/css3_flexbox.asp

Youtube Tutorial

https://www.youtube.com/watch?v=FTIczfR82mQ